LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

UCRL-TR-222074

Multigrid Methods for Mesh Relaxation

Matthew J. O'Brien

June 15, 2006

This work was performed under the auspices of the U.S. Department of Energy by University of California, Lawrence Livermore National Laboratory under Contract W-7405-Eng-48.

Multigrid Methods for Mesh Relaxation.

Math 229B, Final Project, Spring 2006
Due Thursday June 15, 2006
Matthew O'Brien

Abstract

When generating a mesh for the initial conditions for a computer simulation, you want the mesh to be as smooth as possible. A common practice is to use equipotential mesh relaxation to smooth out a distorted computational mesh. Typically a Laplace-like equation is set up for the mesh coordinates and then one or more Jacobi iterations are performed to relax the mesh. As the zone count gets really large, the Jacobi iteration becomes less and less effective and we are stuck with our original unrelaxed mesh. This type of iteration can only damp high frequency errors and the smooth errors remain. When the zone count is large, almost everything looks smooth so relaxation cannot solve the problem. In this paper we examine a multigrid technique which effectively smooths out the mesh, independent of the number of zones.

1 Mesh Relaxation

When generating a mesh for the initial conditions for a computer simulation, it is desirable for the mesh to be as smooth as possible so that your discretized operators can be as accurate as possible and so that you are not needlessly constrained by the Courant-Friedrichs-Levy stability criteria. This problem has been studied since the 1960's when Winslow and Crowley first solved the problem with an equipotential mesh relaxer.

We consider a 2D structured mesh which is topologically equivalent to a square, and has mesh coordinates at the integer lattice points (i, j). We can define generalized coordinates at non-integer values by $\xi = i, j$. The first thing to try is

$$\frac{\partial^2 f}{\partial i^2} + \frac{\partial^2 f}{\partial j^2} = 0$$

where you solve this for f when f is either x or y. This discretizes to

$$-f_{i,j-1} - f_{i-1,j} + 4f_{i,j} - f_{i+1,j} - f_{i,j+1} = 0$$

When using an iterative relaxation scheme this is

$$f_{i,j} = \frac{1}{4}(f_{i,j-1} + f_{i-1,j} + f_{i+1,j} + f_{i,j+1})$$

This is updating a node with the simple average of the node's neighboring nodes. Now we will examine the Winslow-Crowley formulation of the mesh relaxation equation.

The Winslow-Crowley formula is

$$\frac{\partial^2 \xi}{\partial x^2} + \frac{\partial^2 \xi}{\partial y^2} = 0$$

where we solve this two times, once when $\xi = i$ and once when $\xi = j$. Since the information at hand is an array of x and y coordinates as a function of i and j, as opposed to i and j as a function of x and y, we re-write this formula using x and y as a function of i and j. We get the following equation:

$$\alpha_{i,j} \frac{\partial^2 f}{\partial i^2} - 2\beta_{i,j} \frac{\partial^2 f}{\partial i \partial j} + \gamma_{i,j} \frac{\partial^2 f}{\partial j^2} = 0$$

where

$$\begin{aligned}
\gamma &= \left(\frac{\partial x}{\partial i}\right)^2 + \left(\frac{\partial y}{\partial i}\right)^2 \\
&= \frac{1}{4}((x_{i+1,j} - x_{i-1,j})^2 + (y_{i+1,j} - y_{i-1,j})^2) \\
\beta &= \frac{\partial x}{\partial i}\frac{\partial x}{\partial j} + \frac{\partial y}{\partial i}\frac{\partial y}{\partial j} \\
&= \frac{1}{4}((x_{i+1,j} - x_{i-1,j})(x_{i,j+1} - x_{i,j-1}) + (y_{i+1,j} - y_{i-1,j})(y_{i,j+1} - y_{i,j-1}) \\
\alpha &= \left(\frac{\partial x}{\partial j}\right)^2 + \left(\frac{\partial y}{\partial j}\right)^2 \\
&= \frac{1}{4}((x_{i,j+1} - x_{i,j-1})^2 + (y_{i,j+1} - y_{i,j-1})^2)
\end{aligned}$$

This equation can be discretized as

$$\alpha(f_{i+1,j} - 2f_{i,j} + f_{i-1,j}) - \frac{\beta}{2}(f_{i+1,j+1} - f_{i-1,j+1} - f_{i+1,j-1} + f_{i-1,j-1}) + \gamma(f_{i,j+1} - 2f_{i,j} + f_{i,j-1}) = 0$$

Where f is either x or y.

In all previous approaches that I am aware of, this equation was then solved by "successive substitution" which is some form of relaxation, either Jacobi or Gauss-Seidel relaxation. This solution technique is appropriate for low zone count problems or for situations when we do not want to completely relax the mesh to the solution of the mesh relaxation equation, we only want to slightly relax the mesh to locally smooth out the zones. In this paper we are considering mesh generation for the initial conditions for a computer simulation, so we do want the global solution of the mesh relaxation equation to get the best possible mesh as initial conditions for the simulation.

1.1 Test Problems

We have two test problems to show how the Gauss-Seidel relaxation scheme fails to sufficiently relax the mesh as we increase the zone count. The first mesh is a single logical block mesh known as the Kershaw mesh. It starts out with mesh lines making a "Z-pattern". We see how well relaxation can recover the exact solution of the mesh relaxation equation, which is a uniform, orthogonal mesh. The second test problem is a half disc made of 4 logical blocks, with *reduced connectivity* points (a point with 3 zones surrounding it instead of 4) at the corners of the central block. We will look at pictures of the meshes after 100 relaxation iterations for the Kershaw mesh and after 1000 relaxation iterations for the disc mesh. As we increase the zone count, we can see that the relaxed mesh looks more and more like the initial conditions of the mesh, and less and less like the solution of the mesh relaxation equation. This is confirmation of the well known fact that relaxation damps high frequency error modes and is ineffective damping the smooth error. As the zone count increases, everything tends to look smooth so the relaxation becomes less and less effective.

2 Multigrid Solution

In the previous section we have demonstrated that if the exact solution to the mesh relaxation equation is desired, then Gauss-Seidel relaxation is not sufficient for problems with large zone count. To solve this problem, we use a multigrid solver which is specifically designed to damp error modes at all frequencies.

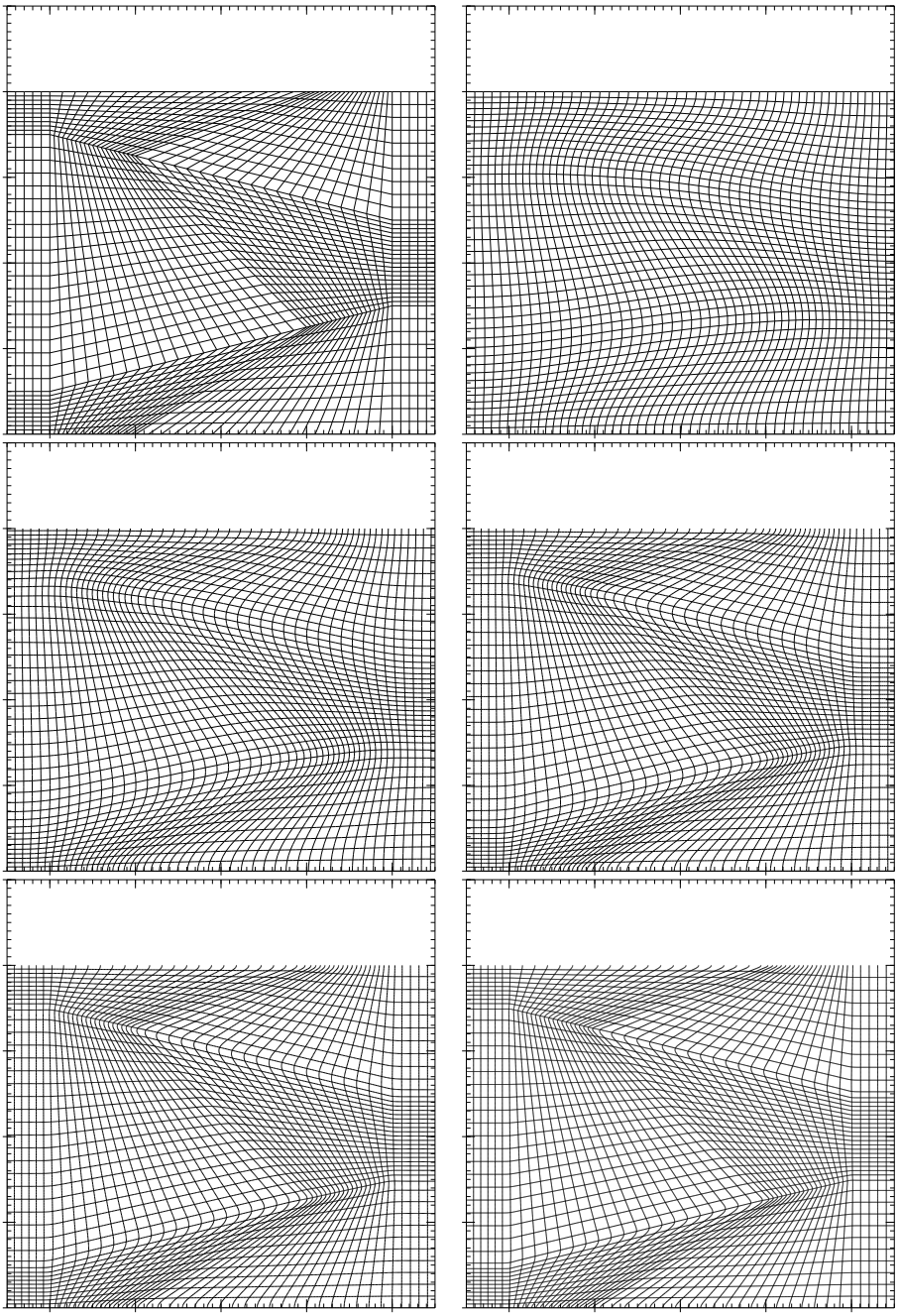

Figure 1: **Relaxation Scaling Study.** The first figure shows the initial conditions of the mesh that we will relax (Kershaw mesh). The other figures show the relaxed mesh after 100 relaxation iterations. The first relaxed mesh has dimensions 50×40 zones and every mesh line is drawn. The second relaxed mesh has dimensions $50 * 2 \times 40 * 2$ zones and only every other mesh line is drawn. The third relaxed mesh has dimensions $50 * 4 \times 40 * 4$ zones and only every fourth mesh line is drawn. The fourth relaxed mesh has dimensions $50 * 8 \times 40 * 8$ zones and only every eighth mesh line is drawn. The fifth relaxed mesh has dimensions $50 * 16 \times 40 * 16$ zones and only every 16^{th} mesh line is drawn. Note that as we have more and more zones, 100 relaxation iterations becomes less and less effective at smoothing out the mesh. Relaxation cannot damp the low frequency modes in the mesh. The exact solution should be a uniform orthogonal mesh.

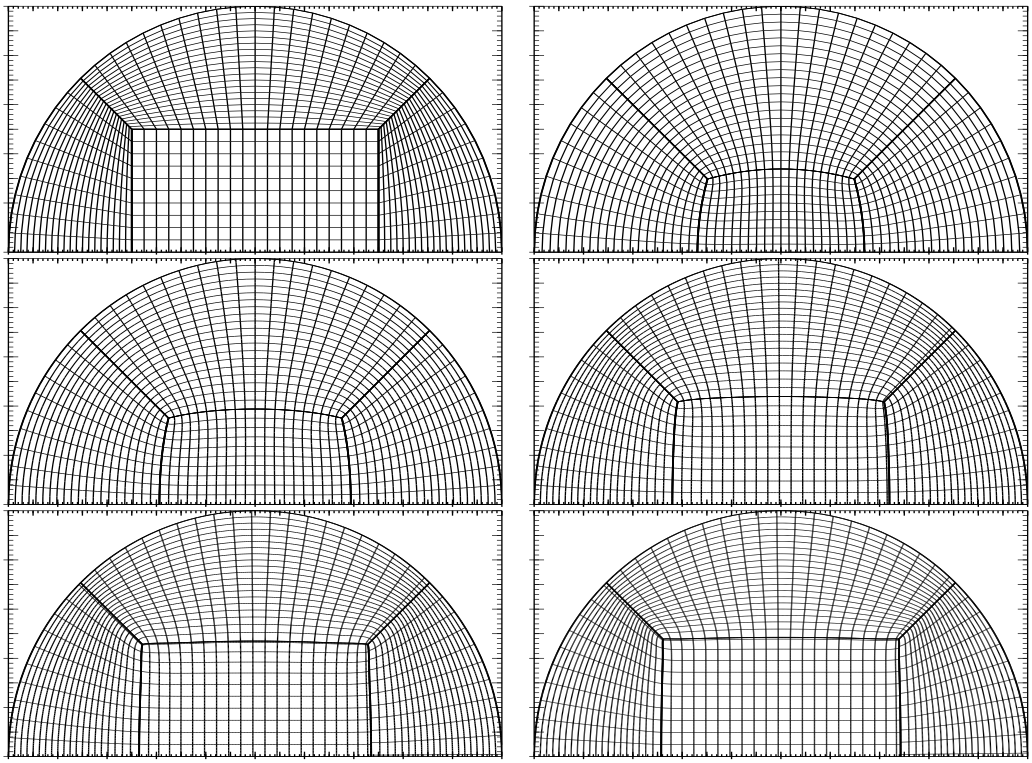

Figure 2: **Relaxation Scaling Study on Circular Mesh.** The first figure shows the initial conditions of the mesh that we will relax. The other figures show the relaxed mesh after 1000 relaxation iterations. The first relaxed mesh has center box dimensions 20×10 zones and the radial blocks have 20 radial zones. Every mesh line is drawn for this coarse mesh. The block boundaries are drawn in bold black lines. In each subsequent picture, we double the number of zones in the radial and angular directions so the total zone count is quadrupled from one picture to the next. So the total number of zones in each picture is $1000, 1000 * 4, 1000 * 4^2, 1000 * 4^3, 1000 * 4^4$. We also do not draw every mesh line since the picture would become completely black with mesh lines. If we have refined a direction by a factor of 2^k, then we draw every $(2^k)^{th}$ mesh line. Note that as the zone count increases, 1000 relaxation iterations are having less and less effect and the mesh looks very similar to the initial conditions. This is because Gauss-Seidel relaxation cannot damp the smooth error components and at such high zone count, almost everything looks smooth. (Note that we are using a different formulation of the mesh relaxation equation for this multi-block problem, other than the Winslow-Crowley formulation. The point of this paper is in examining the *solution technique* of the mesh relaxation equation, not the equation itself, since the same conclusions hold for any Laplace-like mesh relaxation equation).

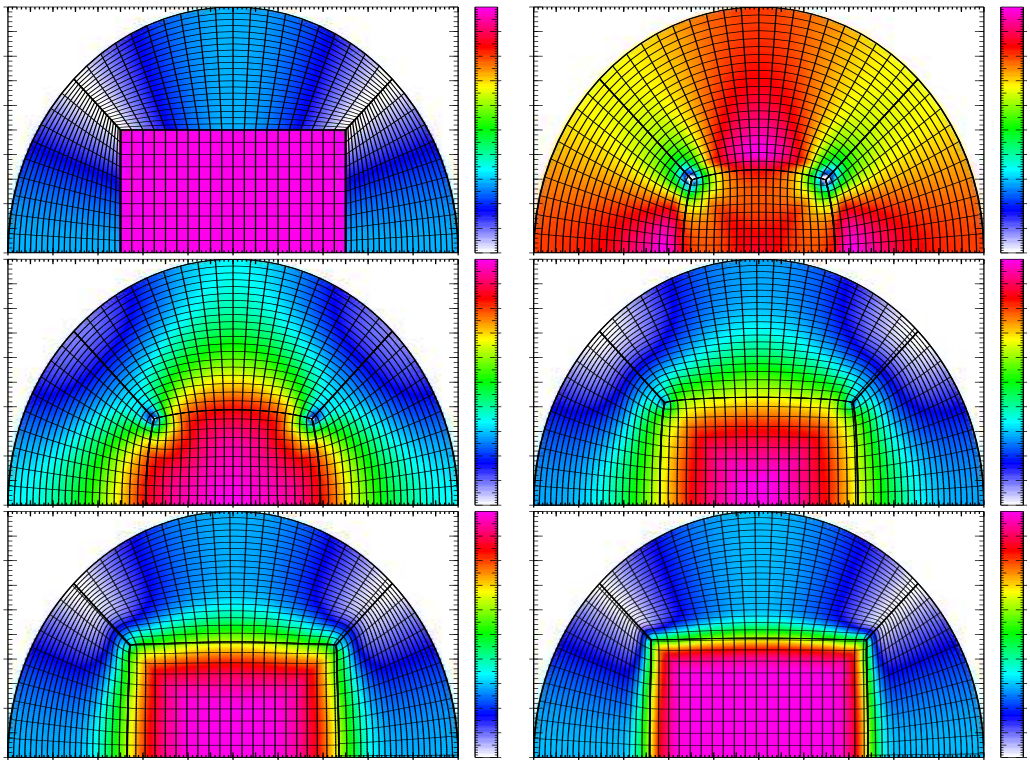

Figure 3: **Zones colored by min edge length.** These figures show pseudo color plots of the zones colored by the min edge length. White and blue mean the zone has an edge with very short length, red means the zone's shortest edge is long. The first figure (upper left) shows the initial conditions of the mesh that we will relax. The next figure (upper right) shows the exact solution of the mesh relaxtion equation. Note that it has a rather uniform distribution of edge lengths, but it is not completely smooth. The other figures show the relaxed mesh after 1000 relaxation iterations. The first relaxed mesh has center box dimensions 20×10 zones and the radial blocks have 20 radial zones. Every mesh line is drawn for this coarse mesh. The block boundaries are drawn in bold black lines. In each subsequent picture, we double the number of zones in the radial and angular directions so the total zone count is quadrupled from one picture to the next. So the total number of zones in each picture is $1000, 1000 * 4, 1000 * 4^2, 1000 * 4^3, 1000 * 4^4$. We also do not draw every mesh line since the picture would become completely black with mesh lines. If we have refined a direction by a factor of 2^k, then we draw every $(2^k)^{th}$ mesh line. Note that as the zone count increases, 1000 relaxation iterations are having less and less effect and the mesh looks very similar to the initial conditions. When we look at the last picture (lower right) we see that it looks very similar to the initial conditions of the mesh, with the center box egde diffused out.

Refinement Factor	Min edge length	Normalized min edge length
1 exact soln	1.000e-1	1.000
1 unrelaxed	5.000e-2	0.500
1	6.297e-2	0.630
2	2.442e-3	0.488
4	1.163e-2	0.465
8	5.747e-3	0.460
16	2.857e-3	0.457

Table 1: **Z-mesh minimum edge lengths.** This table shows the minimum edge lengths of each of the meshes after 100 relaxation iterations for refinement factors of $1, 2, 4, 8, 16$. The first row in the table shows the min edge length of the exact solution of the mesh relaxation equation, which is a uniform orthogonal mesh. The second row in the table shows the min edge length of the initial, un-relaxed mesh. Since we are dividing the zone edge length by 2 as we refine the mesh, we would expect the min edge length to decrease by a factor of 2 each refinement. The "Normalized min edge length" column accounts for this factor, and for the fact that the exact solution of the mesh relaxation equation is a uniform orthogonal mesh, so the exact min edge length should be 0.1, which corresponds to 1.0 in the "Normalized min edge length" column. Note that CFL stability requires that $\Delta t < \Delta x/c$, i.e. the simulation time step must be less than the smallest zone size divided by the wave speed, so needlessly having small zones in your problem is not desirable and makes your simulation take more time steps to finish which takes more wall clock time to finish.

We implemented a multigrid solution via library calls to the HYPRE High Performance Preconditioner library that was developed at Lawrence Livermore National Laboratory as a scalable linear solver. Typically HYPRE is used to solve the diffusion equation in physics packages. But solving this kind of equation is also what is needed for the mesh relaxation problem. Since we want the solver to be able to handle a multi-block mesh with reduced and enhanced connectivity points, we use the BoomerAMG algebraic multigrid solver package within HYPRE.

2.1 The Idea of Multigrid

The idea of multigrid is to use relaxation to beat down the high frequency error on a given mesh, then coarsen the mesh so that what looked smooth on a finer mesh, now looks somewhat more oscillatory. Now relaxation iteration is again applied on the coarser mesh which again beats down high frequency error on this mesh. This is applied recursively to get the multigrid solution.

Suppose that we are trying to solve the equation:

$$Au = f$$

Suppose that we have some approximation u_n, to the exact solution u. Define the *error*, e to be:

$$e = u - u_n$$

that is the error is the exact solution minus the approximate solution. Define the *residual*, r to be:

$$r = f - Au_n$$

Refinement Factor	Min edge length	Normalized min edge length
1 unrelaxed	0.01464	0.66
1	2.204e-2	1.00
2	1.079e-2	0.97
4	4.116e-3	0.74
8	1.910e-3	0.69
16	9.332e-4	0.72

Table 2: **Disc mesh minimum edge lengths.** This table shows the minimum edge lengths of each of the meshes after 1000 relaxation iterations for refinement factors of $1, 2, 4, 8, 16$. The first row in the table shows the min edge length of the initial, un-relaxed mesh. Since we are dividing the zone edge length by 2 as we refine the mesh, we would expect the min edge length to decrease by a factor of 2 each refinement. The "Normalized min edge length" column accounts for this factor. This column also assumes that when the refinement factor is 1 and we have done 1000 relaxation iterations, we have converged to the exact solution of the mesh relaxation equation. This was confirmed by doing several thousand more relaxation iterations and not observing the mesh moving any more. So we define this to have "Normalized min edge length" of 1 and compare the other meshes to this. We can see in this table that as we increase the zone refinement factor, 1000 relaxation iterations become less and less effective. The normalized min edge length decreases (except from 8 to 16). Note that CFL stability requires that $\Delta t < \Delta x/c$, i.e. the simulation time step must be less than the smallest zone size divided by the wave speed, so needlessly having small zones in your problem is not desirable and makes your simulation take more time steps to finish which takes more wall clock time to finish.

Now apply A to the definition of error to get

$$Ae = Au - Au_n = f - Au_n = r$$

So we get the *residual equation*

$$Ae = r$$

So if we could solve the residual equation for e, when $r = 0$, then we would have the exact solution:$u = u_n + e$. So knowing the error is equivalent to knowing the solution. We use multigrid to accurately compute the error of our approximation, so we get an even better approximation.

Define *restriction* to be taking a quantity defined on one mesh and coarsening it down to be defined on a coarser mesh. Let us introduce a superscript notation that indicates the grid spacing the given quantity is defined on. v^h means the quantity v defined on a grid of size h. v^{2h} means the same quantity defined on a coarser mesh of size $2h$. So the restriction operator is such that

$$v^{2h} = restrict(v^h)$$

Restriction can be implemented a variety of different ways such simply dropping components from the fine mesh that do not appear on the coarse mesh, or some other averaging scheme.

Define *interpolation* to be the inverse of restriction. That is, it takes as input a coarse vector v^{2h} and interpolates it onto a finer mesh to produce v^h. This can be implemented as linear interpolation.

$$v^h = interpolate(v^{2h})$$

Now we have the pieces in place to define the **V-cycle multigrid algorithm** (This is from *A Multigrid Tutorial.*).

$$v^h = Vcycle(v^h, f^h)$$

1. Relax $A^h u^h = f^h$ with a given initial guess v^h.
2. If we're on the coarsest grid, then go to step 3.
Else

$$\begin{aligned} r^h &= f^h - A^h v^h \\ r^{2h} &= restrict(r^h) \\ e^{2h} &= 0 \\ e^{2h} &= Vcycle(e^{2h}, r^{2h}) \\ e^h &= Interpolate(e^{2h}) \\ v^h &= v^h + e^h \end{aligned}$$

3. Relax $A^h u^h = f^h$ with initial guess v^h.

From this definition of a Vcycle, we can see that we are trying to solve the residual equation on the coarser grid to get an estimate of the error on the coarse grid, then we interpolate the coarse grid error onto the fine grid to correct our approximate solution. At the end of step 2 we have an accurate solution on the coarser grid, which means have have eliminated the smooth error in the solution, then in step 3, we relax $A^h u^h = f^h$ to eliminate the high frequency error in the solution, at the end of the algorithm, high and low frequency errors have been removed.

A Vcycle is the fundamental algorithm in multigrid. A sequence of V-cycles can be used to construct the **Full Multigrid algorithm**. (This is from *A Multigrid Tutorial*).

$$v^h = FMG(f^h)$$

1. If we're on the coarsest grid, set $v^h = 0$ and goto step 2.
Else

$$\begin{aligned} f^{2h} &= Restrict(f^h) \\ v^{2h} &= FMG(f^{2h}) \\ v^h &= Interpolate(v^{2h}) \end{aligned}$$

2. $v^h = Vcycle(v^h, f^h)$

Note that the procedure $FMG(f^h)$ does not take an initial guess v^h like the Vcycle procedure did. FMG constructs the solution by starting from the coarsest grid and doing V-cycles upward, refining the solution.

3 Conclusions

In this paper we have shown that as zone count increases, solving a mesh relaxation equation by Gauss-Seidel relaxation becomes less effective. Since we are solving a Laplace like equation and we want the exact solution, a multigrid method is an ideal solution. We have implemented a multigrid method using the HYPRE linear solver library. A smooth mesh as initial conditions for a simulation means more accurate

discretized operators and fewer time steps required by CFL stability. For these reasons using multigrid mesh relaxation for generating a mesh for the initial conditions for a simulation is very desirable.

References

[1] Jun, B.I. *A Modified Equipotential Method for Grid Relaxation.* Lawrence Livermore National Laboratory Report, UCRL-JC-138277. March 21, 2000.

[2] Tipton, R. E. *Grid Optimization by Equipotential Relaxation.* Lawrence Livermore National Laboratory Report. July 15, 1992.

[3] Brackbill, J. U. and Saltzman, J. S. *Adaptive Zoning for Singular Problems in Two Dimensions.* J. Comput. Phys. **46,** 342-368 (1982).

[4] *HYPRE High Performance Preconditioners, User's Manual.* Lawrence Livermore National Laboratory Software User's Manual. UCRL-MA-137155 DR. May 30, 2006.

[5] Briggs, W. L., Henson, V. E. and McCormick, S. F. *A Multigrid Tutorial.* SIAM, Philadelphia, PA, 2000.

[6] Trefethen, L. N. and Bau, D. *Numerical Linear Algebra.* SIAM, Philadelphia, PA, 1997.

[7] Demmel, J. W. *Applied Numerical Linear Algebra.* SIAM, Philadelphia, PA, 1997.